Meta-learning how to forecast time series

Thiyanga S Talagala
Department of Statistics, Faculty of Applied Sciences
University of Sri Jayewardenepura, Sri Lanka.
Email: ttalagala@sjp.ac.lk
Corresponding author

Rob J Hyndman
Department of Econometrics and Business Statistics,
Monash University, VIC 3800, Australia.
Email: rob.hyndman@monash.edu

George Athanasopoulos
Department of Econometrics and Business Statistics,
Monash University, VIC 3145, Australia.
Email: george.athanasopoulos@monash.edu

30 November 2022

JEL classification: C10,C14,C22

Meta-learning how to forecast time series

Abstract

Features of time series are useful in identifying suitable models for forecasting. We present a general framework, labelled FFORMS (Feature-based FORecast Model Selection), which selects forecast models based on features calculated from each time series. The FFORMS framework builds a mapping that relates the features of a time series to the "best" forecast model using a classification algorithm such as a random forest. The framework is evaluated using time series from the M-forecasting competitions and is shown to yield forecasts that are almost as accurate as state-of-the-art methods, but are much faster to compute. We use model-agnostic machine learning interpretability methods to explore the results and to study what types of time series are best suited to each forecasting model.

Keywords: Algorithm selection problem, Black-box models, Random forest, Machine learning interpretability, Visualization

1 Introduction

Forecasting is a key activity for many businesses in order to operate efficiently. The rapid advances in computing technologies have enabled businesses to keep track of large numbers of time series. Hence, it is becoming increasingly common to have to regularly forecast many millions of time series. For example, large scale businesses may be interested in forecasting sales, cost, and demand for thousands of products across various locations, warehouses, etc. Technology companies such as Google collect many millions of daily time series such as web-click logs, web search counts, queries, revenues, number of users for different services, etc. Such large collections of time series require fast automated procedures generating accurate forecasts. The scale of these tasks has raised some computational challenges that we seek to address by proposing a new fast algorithm for model selection and time series forecasting. A key component of our work is in studying how this algorithm works, by identifying what types of time series should be forecast with the various available models.

Selecting the most appropriate model for forecasting a given time series can be challenging. Two of the most commonly used automated algorithms are the ETS algorithm of Hyndman et al. (2002) and the automatic ARIMA algorithm of Hyndman & Khandakar (2008). Both algorithms are implemented in the forecast package in R (R Core Team 2018; Hyndman et al. 2021). In this paradigm, a class of models is selected in advance, and many models within that class are estimated for each time series. The model with the smallest AICc value is chosen and used for forecasting. This approach relies on the expert judgement of the forecaster in first selecting the most appropriate class of models to use, as it is not usually possible to compare AICc values between model classes due to differences in the way the likelihood is computed, and the way initial conditions are handled.

An alternative approach, which avoids selecting a class of models *a priori*, is to use a simple "hold-out" test set; but then there is often insufficient data to draw a reliable conclusion. To overcome this drawback, time series cross-validation can be used (Racine 2000; Hyndman & Athanasopoulos 2018); then models from many different classes may be applied, and the model with the lowest cross-validated MSE selected. However, this increases the computation time involved considerably (at least to order n^2 where n is the number of series to be forecast).

Clearly, there is a need for a fast and scalable algorithm to automate the process of selecting models with the aim of forecasting. We refer to this process as forecast-model selection. We propose a general meta-learning framework using features of the time series to select the class of models, or even the specific model, to be used for forecasting. The forecast-model selection process is carried out using a classification algorithm — we use the time series features as inputs, and the "best" forecasting model as the output. The classifier is built using a large historical collection of time series, in advance of the forecasting task at hand. Hence, this is an "offline" procedure.

The "online" process of generating forecasts only involves calculating the features of a time series and using the pre-trained classifier to identify the best forecasting model. Hence, generating forecasts only involves the estimation of a single forecasting model, with no need to estimate large numbers of models within a class, or to carry out a computationally-intensive cross-validation procedure. We refer to this general framework as FFORMS (**F**eature-based **FOR**ecast-**M**odel **S**election).

Currently, there are two methods motivated and using the FFORMS framework as their backbone. Montero-Manso et al. (2020) proposed FFORMA (Feature-based FORecast Model Averaging) which includes, a gradient boosting algorithm with a custom objective function to train the

meta-learner to obtain weights for forecast combinations. FFORMA was ranked as the second most accurate method in the M4-competition. Talagala, Li & Kang (2022) proposed FFORMPP (Feature-based FORecast Model Performance Prediction) which includes efficient Bayesian multivariate surface regression to model forecast error as a function of features. FFORMPP allows rankings of models according to their relative forecasting performance. Both these approaches are considerably slower (and arguably prohibitively slower in an operational sense) than FFORMS. However, the proposed methods show that if we are willing to sacrifice some speed, greater accuracy can be achieved using the FFORMS framework. As we will see in what follows the FFORMS algorithm provides a method that achieves reasonably accurate forecasts within a limited timeline setting.

There have been several recent studies on the use of meta-learning approaches to automate forecast model selection based on features computed from the time series (Shah 1997; Prudêncio & Ludermir 2004; Lemke & Gabrys 2010; Kück, Crone & Freitag 2016). However, to the best of our knowledge, a very limited effort has been made to understand how the meta-learners make their decisions and what is really happening inside these complex model structures. To fill these gaps, this paper makes a first step towards providing a comprehensive analysis of the relationship between time series features and forecast model selection using machine learning interpretability techniques. We try to answer the following questions: i) **What** are the similarities identified by the meta-learner between model classes? ii) **Which** features contribute most to the classification process? iii) **How** are features related to the property being modelled? iv) **How** do features interact with each other to identify a suitable forecasting model? We believe addressing these questions can enhance the transparency of the model predictions and build trust in model's predictions.

The remainder of the paper is structured as follows. We review the related work in Section 2. In Section 3 we explain the detailed components and procedures of our proposed framework for forecast model selection. Section 4 presents the forecasting results in application to the M4-competition data. Section 5 presents the results of what is happening under the hood of FFORMS, followed by some conclusions and suggestions for future work in Section 6.

2 Related work

2.1 Time series features

Reid (1972) points out that the performance of forecasting methods changes according to the nature of the data. Exploring the reasons for these variations may be useful in selecting the

most appropriate model. In response to the results of the M3 competition (Makridakis & Hibon 2000), similar ideas have been put forward by others. Hyndman (2001), Lawrence (2001) and Armstrong (2001) argue that the characteristics of a time series may provide useful insights into which methods are most appropriate for forecasting.

Rather than work with the time series directly at the level of individual observations, we propose analysing time series via an associated "instance space" formed by features of the time series. A time series feature is any measurable characteristic of the time series. For example, Figure 1 (left panel) shows the time-domain representation of six time series taken from the M3 competition (Makridakis & Hibon 2000), while Figure 1 (right panel) shows a feature-based representation of the same time series. Here only two features are considered: the strength of seasonality and the strength of trend, calculated based on the measures introduced by Wang, Smith-Miles & Hyndman (2009). Time series in the lower right quadrant of the scatter plot are non-seasonal but trended, while there is only one series with both high trend and high seasonality. We also see how the degree of seasonality and trend varies between series. Other examples of time series features include autocorrelation, spectral entropy and measures of self-similarity and nonlinearity. Fulcher & Jones (2014) identified 9000 operations to extract features from time series.

The choice of the most appropriate set of features depends on both the nature of the time series being analysed, and the purpose of the analysis. In Section 4, we study the time series from the M1, M3 and M4 competitions (Makridakis et al. 1982; Makridakis & Hibon 2000; Makridakis, Spiliotis & Assimakopoulos 2019), and we select features for the purpose of forecast model selection. The M1, M3 and M4 competitions involve time series of differing length, scale and other properties. We include length as one of our features, but the remaining features are asymptotically independent of the length of the time series (i.e., they are ergodic), and they are independent of scale. As our main focus is forecasting, we select features which have discriminatory power in selecting a good model for forecasting.

2.2 Meta-learning for algorithm selection

John Rice was an early and strong proponent of the idea of meta-learning, which he called the algorithm selection problem (ASP) (Rice 1976). The term *meta-learning* started to appear with the emergence of the machine-learning literature.

Rice's framework for algorithm selection is shown in Figure 2 and comprises four main components. The problem space, P, represents the data sets used in the study. The feature space, F, is the range of measures that characterize the problem space P. The algorithm space, A, is

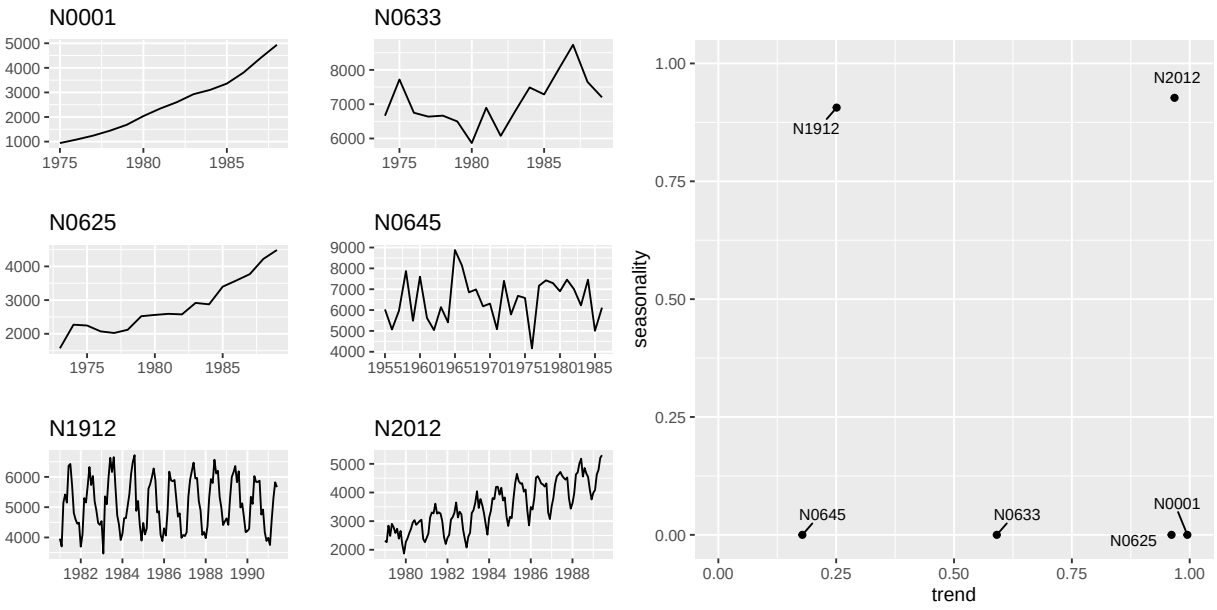

Figure 1: *Time-domain representation of time series (left) and feature-based representation of time series (right).*

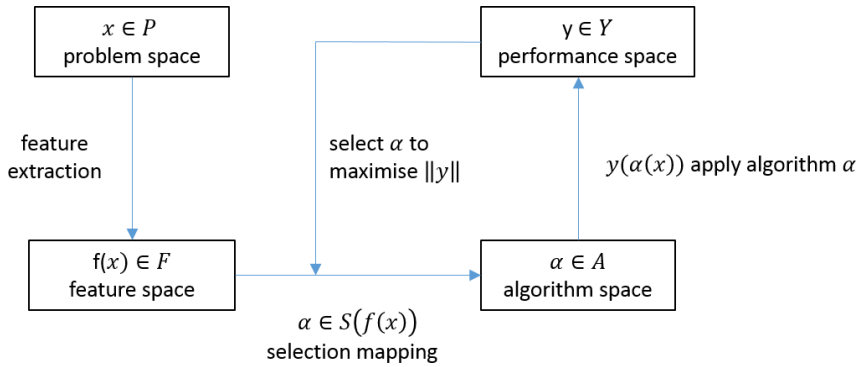

Figure 2: *Rice's framework for the Algorithm Selection Problem.*

a list of suitable candidate algorithms which can be used to find solutions to the problems in P. The performance metric, Y, is a measure of algorithm performance such as accuracy, speed, etc. The main challenge in ASP is to identify the selection mapping S from the feature space to the algorithm space A. A formal definition of the algorithm selection problem is given by Smith-Miles (2009), and repeated below.

> **Algorithm selection problem**. For a given problem instance $x \in P$, with features $f(x) \in F$, find the selection mapping $S(f(x))$ into algorithm space A, such that the selected algorithm $\alpha \in A$ maximizes the performance mapping $y(\alpha(x)) \in Y$.

2.3 Forecast-model selection using meta-learning

Selecting models for forecasting can be framed according to Rice's ASP framework.

> **Forecast-model selection problem**. For a given time series $x \in P$, with features $f(x) \in F$, find the selection mapping $S(f(x))$ into the algorithm space A, such that

the selected algorithm $\alpha \in A$ minimizes forecast accuracy error metric $y(\alpha(x)) \in Y$ on the test set of the time series.

The training set is used to estimate the parameters of a forecasting model. Based on this fitted model, we generate forecasts over the test set and compare them against the test set values. Since the test data are not used for model fitting, this practice provides a reliable indicator to evaluate how well the model makes accurate forecasts on new data.

Existing methods differ with respect to the way they define the problem space (A), the features (F), the forecasting accuracy measure (Y) and the selection mapping (S).

Collopy & Armstrong (1992) introduced 99 rules based on 18 features of time series, in order to make forecasts for economic and demographic time series. This work was extended by Armstrong (2001) to reduce human intervention.

Shah (1997) used the following features to classify time series: the number of observations, the ratio of the number of turning points to the length of the series, the ratio of the number of step changes, skewness, kurtosis, the coefficient of variation, autocorrelations at lags 1–4, and partial autocorrelations at lags 2–4. Casting Shah's work in Rice's framework, we can specify: $P = 203$ quarterly series from the M1 competition (Makridakis et al. 1982); $A = 3$ forecasting methods, namely simple exponential smoothing, Holt-Winters exponential smoothing with multiplicative seasonality, and a basic structural time series model; $Y =$ mean squared error for a hold-out sample. Shah (1997) learned the mapping S using discriminant analysis.

Prudêncio & Ludermir (2004) was the first paper to use the term "meta-learning" in the context of time series model selection. They studied the applicability of meta-learning approaches for forecast model selection based on two case studies. Again using the notation above, we can describe their first case study with: A contained only two forecasting methods, simple exponential smoothing and a time-delay neural network; $Y =$ mean absolute error; F consisted of 14 features, namely length, autocorrelation coefficients, coefficient of variation, skewness, kurtosis, and a test of turning points to measure the randomness of the time series; S was learned using the C4.5 decision tree algorithm. For their second study, the algorithm space included a random walk, Holt's linear exponential smoothing and AR models; the problem space P contained the yearly series from the M3 competition (Makridakis & Hibon 2000); F included a subset of features from the first study; and Y was a ranking based on error. Beyond the task of forecast-model selection, they used the NOEMON approach to rank the algorithms (Kalousis & Theoharis 1999).

Lemke & Gabrys (2010) studied the applicability of different meta-learning approaches for time series forecasting. Their algorithm space A contained ARIMA models, exponential smoothing models and a neural network model. In addition to statistical measures such as the standard deviation of the de-trended series, skewness, kurtosis, length, strength of trend, Durbin-Watson statistics of regression residuals, the number of turning points, step changes, a predictability measure, nonlinearity, the largest Lyapunov exponent, and auto-correlation and partial-autocorrelation coefficients, they also used frequency domain based features. The feed forward neural network, decision tree and support vector machine approaches were considered to learn the mapping S.

Wang, Smith-Miles & Hyndman (2009) used a meta-learning framework to provide recommendations as to which forecast method to use to generate forecasts. In order to evaluate forecast accuracy, they introduced a new measure $Y =$ *simple percentage better (SPB)*, which calculates the forecasting accuracy of a method against the forecasting accuracy error of a random walk model. They used a feature set F comprising nine features: strength of trend, strength of seasonality, serial correlation, nonlinearity, skewness, kurtosis, self-similarity, chaos and periodicity. The algorithm space A included eight forecast models, namely, exponential smoothing, ARIMA, neural networks and random walk model; while the mapping S was learned using the C4.5 algorithm for building decision trees. In addition, they used SOM clustering on the features of the time series in order to understand the nature of time series in a two-dimensional setting.

The set of features introduced by Wang, Smith-Miles & Hyndman (2009) was later used by Widodo & Budi (2013) to develop a meta-learning framework for forecast-model selection. The authors further reduced the dimensionality of time series by performing principal component analysis on the features.

More recently, Kück, Crone & Freitag (2016) proposed a meta-learning framework based on neural networks for forecast-model selection. Here, $P = 78$ time series from the NN3 competition were used to build the meta-learner. They introduced a new set of features based on forecasting errors. The average symmetric mean absolute percentage error was used to identify the best forecast models for each series. They classify their forecast models in the algorithm space A, comprising single, seasonal, seasonal-trend and trend exponential smoothing. The mapping S was learned using a feed-forward neural network. Further, they evaluated the performance of different sets of features for forecast-model selection.

3 Methodology

3.1 FFORMS framework

Our proposed FFORMS framework, presented in Figure 3, builds on this preceding research. The offline and online phases are shown in blue and red respectively. A classification algorithm (the meta-learner) is trained during the offline phase and is then used to select an appropriate forecast model for a new time series in the online phase. We use machine learning interpretability tools to gain insights into what is happening under the hood of the FFORMS framework.

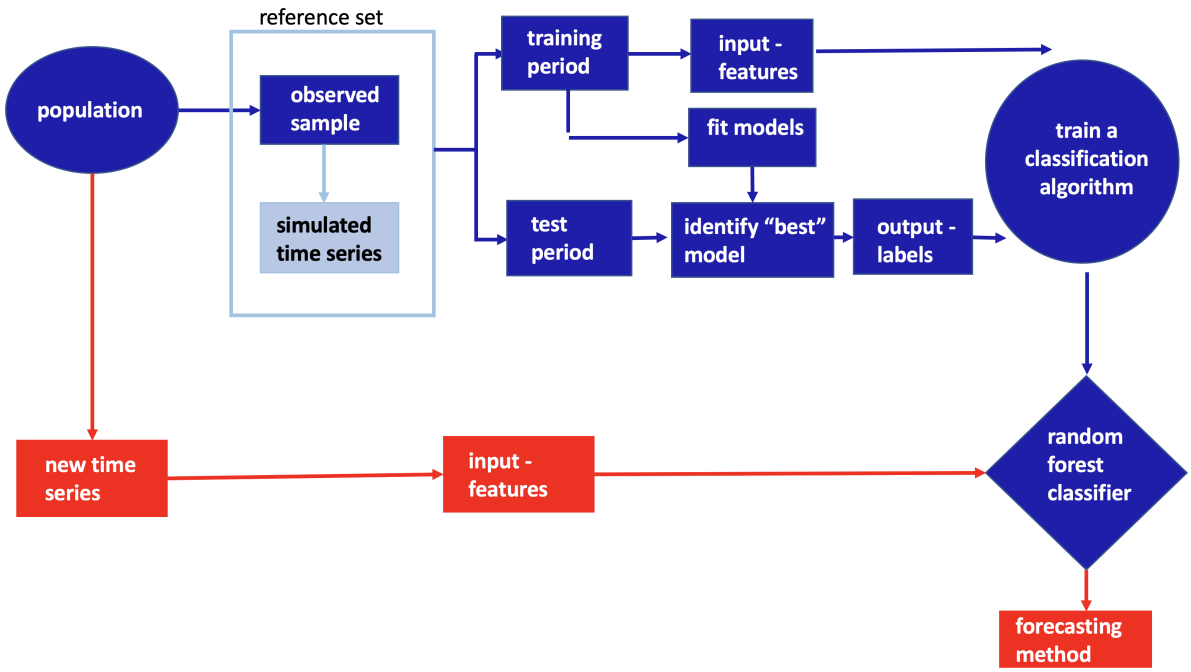

Figure 3: *FFORMS (Feature-based FORecast-Model Selection) framework. The offline phase is shown in blue, the online phase is in red.*

In order to train our classification algorithm, we need a large collection of time series which are similar to those we will be forecasting. We assume that we have an essentially infinite population of time series, and we take a sample of them in order to train the classification algorithm denoted as the "observed sample". The new time series we wish to forecast can be thought of as additional draws from the same population. Hence, any conclusions made from the classification framework refer only to the population from which the sample has been selected. We may call this the "target population" of time series. It is important to have a well-defined target population to avoid misapplying the classification rules. In practice, we may wish to augment the set of observed time series by simulating new time series similar to those in the assumed population (we provide details and discussion in Section 3.2 that follows). We denote the total collection of time series used for training the classifier as the "reference set".

Each time series within the reference set is split into a training period and a test period. From each training period we compute a range of time series features, and fit a selection of candidate models. The calculated features form the input vector to the classification algorithm. Using the fitted models, we generate forecasts and identify the "best" model for each time series based on a forecast error measure (e.g., MASE) calculated over the test period. The models deemed "best" form the output labels for the classification algorithm. The pseudo code for our proposed framework is presented in Algorithm 1 below. In the following sections, we briefly discuss aspects of the offline phase.

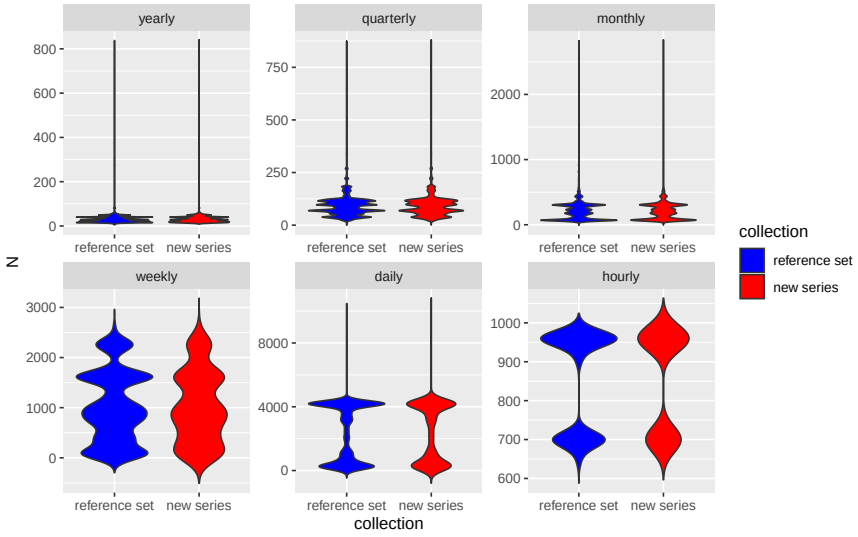

Figure 4: *Distribution of lengths of the series in the reference set and new series collections.*

3.2 Augmenting the observed sample with simulated time series

In practice, we may wish to augment the set of observed time series by simulating new time series similar to those in the assumed population. This process may be useful when the number of observed time series is too small to build a reliable classifier. Alternatively, we may wish to add more of some particular types of time series to the reference set in order to get a more balanced sample for the classification.

There are several options that could be used to produce simulated series that are similar to those in the population. One approach is to fit models to the time series in the observed sample, and then use those models as data generating processes to obtain similar time series. This could be done, for example, using exponential smoothing models, ARIMA models or other modelling functions in the forecast package in R (Hyndman et al. 2021). More specifically, suppose the for the first time in the observed sample the ARIMA model selected from the automated ARIMA (auto.arima) in the forecast package is $y_t = 5 + 0.52\epsilon_{t-1}$. Then, from this fitted model, we can simulate multiple time series. Similarly, we can fit models for all the time series in the observed

Algorithm 1 The FFORMS framework - Forecasting based on meta-learning.

Offline phase - train the classifier

Given:

$O = \{x_1, x_2, \ldots, x_n\}$: the collection of n observed time series.
L : the set of class labels (e.g. ARIMA, ETS, SNAIVE).
F : the set of functions to calculate time series features.
nsim : number of series to be simulated.
B : number of trees in the random forest.
k : number of features to be selected at each node.

Output:

FFORMS classifier

Prepare the reference set

1: For $i = 1$ to n:
2: Simulate *nsim* time series similar to x_i, using either a model or features to determine the data generating process.
3: The time series in O and simulated time series from step 1 form the reference set $R = \{x_1, x_2, \ldots, x_n, x_{n+1}, \ldots, x_N\}$ where $N = n + nsim$.
4: EndFor.

Prepare the meta-data

5: For $j = 1$ to N:
6: Split x_j into a training period and test period.
7: Calculate features F based on the training period.
8: Fit L models to the training period.
9: Calculate forecasts for the test period from each model.
10: Calculate forecast error measure over the test period for all models in L.
11: Select the model with the minimum forecast error.
12: Meta-data: input features (step 4), output labels (step 8).
13: EndFor.

Train a random forest classifier

14: Train a random forest classifier based on the meta-data giving an ensemble of trees $\{T_b\}, b = 1, \ldots, B$.

Online phase - forecast a new time series

Given:

a new time series x_{new}.
FFORMS classifier from step 10.

Output:

a class label for x_{new}.

15: Calculate features of x_{new}.
16: Let $\hat{C}_b(x_{new})$ be the class prediction of the b^{th} random forest tree. Then the class label for x_{new} is $\hat{C}_{rf}(x_{new}) = \text{majorityvote}\{\hat{C}_b(x_{new})\}_1^B$.

sample to find models to simulate time series. Assuming the models produce data that are similar to the observed time series ensures that the simulated series can be considered part of the time series population. An alternative approach is to generate new time series with similar *features* from those found in the reference set (Kang, Hyndman & Li 2020), using the gratis package in R (Kang et al. 2020). Either way, these simulated series are produced in the offline phase of the algorithm, so the computational time in producing them is of no real consequence.

3.3 Input: features

Although the FFORMS framework could use any time series features, our current implementation uses the features shown in Table 1; these are defined in the Appendix. An attribute of the proposed FFORMS framework is that its online phase is fast compared to the time required for

implementing a typical model selection or cross-validation procedure. Therefore, we consider only features that can be computed rapidly, as this computation must be done during the online phase. Furthermore, all our features are easily interpretable.

Table 1: *Description of time series features used in FFORMS*

	Feature	Description	Y	Q/M	W	D/H
1	T	length of time series	✓	✓	✓	✓
2	trend	strength of trend	✓	✓	✓	✓
3	seasonality_q	strength of quarterly seasonality	-	✓	-	-
4	seasonality_m	strength of monthly seasonality	-	✓	-	-
5	seasonality_w	strength of weekly seasonality	-	-	✓	✓
6	seasonality_d	strength of daily seasonality	-	-	-	✓
7	seasonality_y	strength of yearly seasonality	-	-	-	✓
8	linearity	linearity	✓	✓	✓	✓
9	curvature	curvature	✓	✓	✓	✓
10	spikiness	spikiness	✓	✓	✓	✓
11	e_acf1	first ACF value of remainder series	✓	✓	✓	✓
12	stability	stability	✓	✓	✓	✓
13	lumpiness	lumpiness	✓	✓	✓	✓
14	entropy	spectral entropy	✓	✓	✓	✓
15	hurst	Hurst exponent	✓	✓	✓	✓
16	nonlinearity	nonlinearity	✓	✓	✓	✓
17	alpha	ETS(A,A,N) $\hat{\alpha}$	✓	✓	✓	-
18	beta	ETS(A,A,N) $\hat{\beta}$	✓	✓	✓	-
19	hwalpha	ETS(A,A,A) $\hat{\alpha}$	-	✓	-	-
20	hwbeta	ETS(A,A,A) $\hat{\beta}$	-	✓	-	-
21	hwgamma	ETS(A,A,A) $\hat{\gamma}$	-	✓	-	-
22	ur_pp	test statistic based on Phillips-Perron test	✓	-	-	-
23	ur_kpss	test statistic based on KPSS test	✓	-	-	-
24	y_acf1	first ACF value of the original series	✓	✓	✓	✓
25	diff1y_acf1	first ACF value of the differenced series	✓	✓	✓	✓
26	diff2y_acf1	first ACF value of the twice-differenced series	✓	✓	✓	✓
27	y_acf5	sum of squares of first 5 ACF values of original series	✓	✓	✓	✓
28	diff1y_acf5	sum of squares of first 5 ACF values of differenced series	✓	✓	✓	✓
29	diff2y_acf5	sum of squares of first 5 ACF values of twice-differenced series	✓	✓	✓	✓
30	sediff_acf1	ACF value at the first lag of seasonally-differenced series	-	✓	✓	✓
31	sediff_seacf1	ACF value at the first seasonal lag of seasonally-differenced series	-	✓	✓	✓
32	sediff_acf5	sum of squares of first 5 autocorrelation coefficients of seasonally-differenced series	-	✓	✓	✓
33	seas_pacf	partial autocorrelation coefficient at first seasonal lag	-	✓	✓	✓
34	lmres_acf1	first ACF value of residual series of linear trend model	✓	-	-	-
35	y_pacf5	sum of squares of first 5 PACF values of original series	✓	✓	✓	✓
36	diff1y_pacf5	sum of squares of first 5 PACF values of differenced series	✓	✓	✓	✓
37	diff2y_pacf5	sum of squares of first 5 PACF values of twice-differenced series	✓	✓	✓	✓

3.4 Output: labels

The task of the classification algorithm is to identify the "best" forecasting method for a given time series. The candidate models considered as labels will depend on the observed time series. For example, if we have only non-seasonal time series, and no chaotic features, we may wish to restrict the models to white noise, random walk, ARIMA and ETS processes. We fit the corresponding models outlined in Table 2 to each series in the reference set.

Each candidate model considered is estimated strictly on the training period of each series in the reference set. Forecasts are then generated for the test period over which the chosen forecast accuracy measure is calculated. The model with the lowest forecast error measure over the test period is deemed "best". This step is the most computationally intensive and time-consuming, as each candidate model has to be applied on each series in the reference set. However, as this task is done during the offline phase of the FFORMS framework, the time involved and computational cost associated are of no real significance and are completely controlled by the user.

For each series in the reference set and each forecasting model, we compute the accuracy measures that were used in the M4-competition (Makridakis, Spiliotis & Assimakopoulos 2019), namely the MASE and the symmetric mean absolute percentage error (sMAPE). Each of these is standardised by the median MASE and median sMAPE, calculated across the forecast models. The model with the lowest average value of the scaled MASE and scaled sMAPE is selected as the output class label.

Table 2: *Description of class labels used in FFORMS*

Class label	Description	Y	Q/M	W	D/H
wn	white noise process	✓	✓	✓	✓
ARMA	AR, MA, ARMA processes	✓	✓	✓	-
ARIMA	ARIMA process	✓	✓	✓	-
SARIMA	seasonal ARIMA	-	✓	✓	-
rwd	random walk with drift	✓	✓	✓	✓
rw	random walk	✓	✓	✓	✓
theta	standard theta method	✓	✓	✓	✓
stlar	seasonal decomposition with AR modelling of the seasonally adjusted series	-	✓	✓	✓
ETS_NTNS	ETS without trend and seasonal components	✓	✓	-	-
ETS_T	ETS with trend component and without seasonal component	✓	✓	-	-
ETS_DT	ETS with damped trend component and without seasonal component	✓	✓	-	-
ETS_TS	ETS with trend and seasonal component	-	✓	-	-
ETS_DTS	ETS with damped trend and seasonal component	-	✓	-	-
ETS_S	ETS with seasonal component and without trend component	-	✓	-	-
snaive	seasonal naive method	-	✓	✓	✓
tbats	TBATS forecasting	-	✓	✓	✓
nn	neural network time series forecasts	✓	✓	✓	✓
mstlets	multiple seasonal decomposition with ETS modelling of the seasonally adjusted series	-	-	✓	✓
mstlarima	multiple seasonal decomposition with ARIMA modelling of the seasonally adjusted series	-	-	-	✓

3.5 Train a FFORMS meta-learner

In principle, any classification algorithm could be used to train the meta-learner. Our current implementation uses a random forest (RF) algorithm because: i) it can model complex interactions between features; ii) the modelling framework captures linear and non-linear effects of features through the averaging of large number of decision trees; iii) it is robust against over-fitting the

training data; iv) it is easy to handle the problem of imbalanced classes; v) it is a fast approach compared to boosting algorithms; and vi) it is fairly easy and straightforward to implement with available software. In this work, we have used the randomForest package (Liaw & Wiener 2002; Breiman et al. 2018) which implements the Fortran code for random forest classification by Breiman & Cutler (2004).

To determine the class label for a new instance, features are calculated and passed down the trees. Then each tree gives a prediction and the majority vote over all individual trees leads to the final decision.

The RF algorithm is highly sensitive to class imbalance (Breiman 2001), and our reference set is unbalanced: some classes contain significantly more cases than other classes. The degree of class imbalance is reduced to some extent by augmenting the observed sample with the simulated time series. We consider three approaches to address the class imbalance in the data: (1) incorporating class priors into the RF classifier; (2) using the balanced RF algorithm introduced by Chen, Liaw & Breiman (2004); and (3) re-balancing the reference set with down-sampling. In down-sampling, the size of the reference set is reduced by down-sampling the larger classes so that they match the smallest class in size; this potentially discards some useful information. Comparing the results, the balanced RF algorithm and RF with down-sampling did not yield satisfactory results. We therefore only report the results obtained by the RF built on unbalanced data (RF-unbalanced) and the RF with class priors (RF-class priors). The RF algorithms are implemented by the randomForest R package (Liaw & Wiener 2002; Breiman et al. 2018). The likelihood that a given class will appear in the bootstrap samples is represented by the class priors. The class priors are introduced through the option `classwt`. We use the reciprocal of class size as the class priors. The number of trees `ntree` is set to 1000, and the number of randomly selected features `k` is set to be one third of the total number of features available and the minimum number of leaf node observations is 1. The number of trees is set to 1000 to reduce the time in both the online and offline phases of the algorithm.

Our aim is different from most classification problems in that we are not interested in accurately predicting the class, but in finding the best possible forecast model. It is possible that two models produce almost equally accurate forecasts, and therefore it is not important whether the classifier picks one model over the other. Therefore we report the forecast accuracy obtained from the FFORMS framework, rather than the classification accuracy.

4 Application to M4-competition data

4.1 Evaluation choices

To test how well our proposed framework can identify suitable models for forecasting, we use the time series from the M1 (Makridakis et al. 1982), M3 (Makridakis & Hibon 2000) and M4 (Makridakis, Spiliotis & Assimakopoulos 2019) competitions. The data from the M-competitions are a sample of time series collected from several domains such as demography, finance, business and economics. In our experiment, we treat the time series from the M1 and M3 competitions as the observed sample. We augment these by adding multiple time series simulated using the training period of each series in the M4-competition to the reference set. For this purpose, we fit models to each series, and simulate new time series from those models. For, yearly, quarterly and monthly data, we use ETS and ARIMA models via the `ets()` and `auto.arima()` functions in the forecast package. For weekly, daily and hourly series, we use a multiple seasonal decomposition (MSTL) (Bandara, Hyndman & Bergmeir 2021), and forecast the seasonally adjusted series using ETS models. This is implemented in the `stlf()` function from the forecast package.

We build separate FFORMS meta-learners for yearly, quarterly, monthly, daily, and hourly series, due to their differences in features and class labels. Then the pre-trained FFORMS meta-learners are used to forecast the test period of the M4-competition time series.

4.2 Summary of the main results

Table 3 shows the performance of the FFORMS approach on the M4-competition data. The point forecasts and prediction intervals are evaluated based on the test period of each series. The point forecasts are evaluated based on the MASE computed for each forecast horizon, and then by averaging the MASE errors across all series corresponding to each frequency category. Similarly, the mean scaled interval score MSIS (Makridakis, Spiliotis & Assimakopoulos 2019) is used to evaluate the prediction intervals. The results are compared against several benchmarks and the top three ranked methods of the M4-competition. The top-ranking methods of the M4-competition were based on some type of combination approach. The first ranked method was a hybrid approach that produced forecasts based on both ETS and machine learning approaches. The second and third places were based on a combination of nine and seven different forecast models. Recall that the second ranked approach was FFORMA which is based on FFORMS. All these approaches are time-consuming as forecasts from all candidate models must be computed. The FFORMS results are based on forecasts from a single selected model.

Table 3: *The performance of FFORMS on the M4-competition data based on point forecasts (based on MASE) and prediction intervals (based on MSIS)*

Point Forecasts (Mean Absolute Scaled Error (MASE))						
	Yearly	Quarterly	Monthly	Weekly	Daily	Hourly
FFORMS	3.17	1.20	0.98	2.31	3.57	0.84
auto.arima	3.40	1.17	0.93	2.55	-	-
ets	3.44	1.16	0.95	-	-	-
theta	3.37	1.24	0.97	2.64	3.33	1.59
rwd	3.07	1.33	1.18	2.68	3.25	11.45
rw	3.97	1.48	1.21	2.78	3.27	11.60
nn	4.06	1.55	1.14	4.04	3.90	1.09
stlar	-	2.02	1.33	3.15	4.49	1.49
snaive	-	1.66	1.26	2.78	24.46	2.86
tbats	-	1.19	1.05	2.49	3.27	1.30
wn	13.42	6.50	4.11	49.91	38.07	11.68
mstlarima	-	-	-	-	3.84	1.12
mstlets	-	-	-	-	3.73	1.23
combination (mean)	4.09	1.58	1.16	6.96	7.94	3.93
M4-1st	**2.98**	1.12	**0.88**	2.36	3.45	0.89
M4-2nd	3.06	**1.11**	0.89	**2.11**	3.34	**0.81**
M4-3rd	3.13	1.12	0.91	2.16	**2.64**	0.87
Prediction Intervals (Mean Scaled Interval Score (MSIS))						
FFORMS	39.79	11.24	9.82	**20.84**	36.36	8.0
M4-1st	**23.89**	**8.55**	**7.20**	22.03	**26.28**	**7.92**
M4-2nd	27.47	9.38	8.65	21.53	34.37	18.50
M4-3rd	not submitted					
naive	56.55	14.07	12.30	26.35	32.55	71.24

Note: Bold signifies the best performing method.

Table 4: *Computational time for producing forecasts based on 100 randomly selected series from each frequency category of the M4-competition data set.*

Computational time for producing forecasts in seconds (IQR)						
	Yearly	Quarterly	Monthly	Weekly	Daily	Hourly
FFORMS	3.38 (0.18)	20.98 (8.23)	100.13 (7.13)	128.41 (5.17)	77.20 (5.67)	34.53 (5.16)
auto.arima	5.91 (0.05)	42.11 (2.15)	448.41 (1.95)	584.98 (2.35)	-	-
ets	1.14 (0.02)	16.92 (0.09)	115.50 (0.17)	-	-	-
theta	2.74 (2.48)	10.15 (11.45)	29.13 (1.15)	96.06 (0.42)	83.77 (2.67)	54.32 (2.32)
rwd	0.29 (5.42)	0.29 (8.20)	0.33 (15.57)	0.34 (21.78)	0.41 (33.72)	0.37 (26.46)
rw	0.16 (4.65)	0.20 (6.67)	0.26 (15.68)	0.22 (17.16)	0.27 (19.56)	0.24 (9.58)
nn	2.32 (0.14)	6.54 (0.23)	32.78(0.25)	281.68 (0.61)	424.28 (1.71)	354.97 (3.61)
stlar	-	0.83 (17.97)	0.94 (12.03)	0.90 (10.11)	2.21 (0.12)	1.70 (0.01)
snaive	-	0.18 (4.51)	0.30 (3.12)	0.20 (1.44)	0.32 (0.34)	0.44 (2.12)
tbats	-	20.16 (6.98)	38.12 (2.44)	40.16 (3.36)	68.73 (0.65)	49.52 (2.98)
wn	0.19 (2.65)	0.20 (4.30)	0.23 (0.08)	0.19 (4.51)	0.26 (1.00)	0. 22 (0.05)
mstlarima	-	-	-	-	86.92 (0.52)	30.60 (0.17)
mstlets	-	-	-	-	19.79 (0.09)	10.13 (0.48)

According to the Table 3, we can see the FFORMS yields relatively accurate forecasts comparable to benchmarks and the top-3 of the M4-competition. The purpose of FFORMS framework is not to introduce an algorithm that beats all benchmark approaches but to introduce an algorithm that performs reasonably well across a wide range of time series, and which is relatively fast to implement. There is usually a trade-off when forecasting between accuracy and computation time, and FFORMS is intended to trade off some accuracy for large gains in computational

efficiency. It gives results that are relatively similar to the best methods in the M4-competition, while being orders of magnitude faster to apply in the online phase.

4.3 Computation times

The running time for fitting models are reported in Table 4. The reported values are based on 100 randomly selected time series from each frequency category of M4-competition data. The reported results are median time along with IQR based on 100 replicates using 24 core Xeon-E5-2680-v3@2.50GHz servers. The microbenchmark (Mersmann 2021) R package was used for calculations.

The times to produce the forecasts that were in the top few places of the M4-competition are not shown, as these were provided to use in the M4 results. However, the times are reported in Makridakis, Spiliotis & Assimakopoulos (2019), albeit with a different computing archictecture, and FFORMS is much faster than any of the competitor methods shown in Table 3.

5 Peeking inside FFORMS

Apart from providing a fast forecasting method that has relatively good performance, we also wish to better understand what methods work best for different types of time series.

To address this problem, we present the results for yearly series (representing non-seasonal time series), monthly series (representing time series with a single seasonal component), and hourly series (representing series with multiple seasonal components). The results for quarterly and weekly data were very similar to the corresponding results for monthly data, while the results for daily data were very similar to the results for hourly data. Hence, these are omitted.

5.1 Visualization of the similarity between forecast models

In FFORMS, the forecast-model selection problem is posed as a supervised learning task. Each time series in the meta-data set is represented as a vector of features and we compute the probability of each model being the "best" forecast model. Sometimes it is clear that one model is better than the others, and it takes probability close to one. But for other series, there might be two almost equally good models, and these would then take similar probabilities. Because we are interested only in obtaining good forecast accuracy, and not in identifying the "best" model, this is of no concern. These probabilities are obtained from the vote matrix which contains one row for each time series and one column for each model (or class label). Each cell contains the fraction of trees in the forest that classified each time series to each class. We use a vote matrix calculated based on out-of-bag observations. We can study these vote-matrix probabilities for

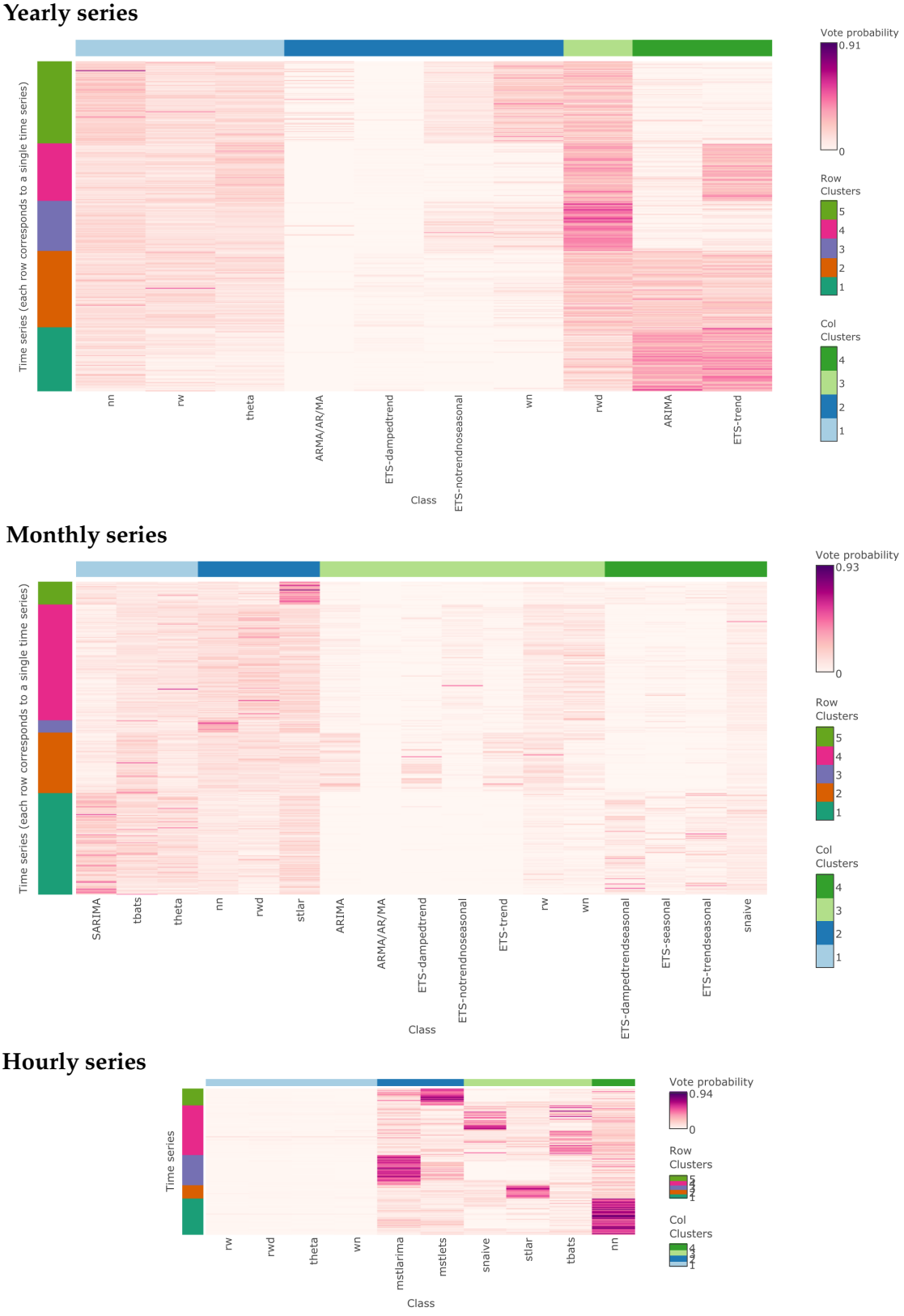

Figure 5: *Heatmaps of the vote matrices for yearly, monthly and hourly series. Each cell shows the number of votes received by the time series in that row for the forecast model in that column.*

Table 5: *Summary of column clusters displayed in vote matrices of Figure 5.*

Yearly	Monthly	Hourly
Cluster 1 **Models flexible for any situation**	*Cluster 1* **Models flexible for any situation**	*Cluster 1* **Models without seasonal component and Theta method**
Neural network Random walk Theta	SARIMA tbats Theta	Random walk Random walk with drift Theta White noise process
Cluster 2 **Models not suitable to handle highly trended series** ARMA/AR/MA ETS with damped trend ETS without trend and seasonality White noise process	*Cluster 2* **Models that can handle series with complex patterns** Neural networks Random walk with drift STLAR	*Cluster 2* **Models that can handle multiple seasonality** mstlarima mstlets
Cluster 3 **Random walk with drift** Random walk with drift	*Cluster 3* **Models without seasonal component** ARIMA ARMA/AR/MA ETS-damped trend ETS without trend and seasonality ETS with trend Random walk White noise process	*Cluster 3* **Models for time varying seasonality** snaive STLAR TBATS
Cluster 4 **Models that can handle highly trended series** ARIMA ETS with trend	*Cluster 4* **ETS models with seasonal component and snaive** ETS with damped trend and seasonality ETS with seasonal component ETS with trend and seasonality snaive	*Cluster 4* **Neural network** Neural network

all time series in order to explore FFORMS's understanding of the similarities between forecast models.

Figure 5 shows heatmaps (Schep & Kummerfeld 2017) of the vote matrices, with rows and columns reordered and clustered using a hierarchical clustering approach. Each cell in the plots displays the number of votes received by the time series in that row over the candidate forecast-model in that column. We can see the columns are clustered according to the similarities of the forecasting models. The results are summarized in Table 5, with an interpretation of each cluster given in bold text. For the yearly results, the models are clustered according to their ability to handle trends. For monthly and hourly series, the models are clustered according to the types of seasonal and trend components they can handle. This suggests that our meta-learner has correctly identified some similarities between model classes.

Some more subtle insights also emerge:

1. Regardless of the true class label (corresponding to the model that gives the best forecasts on the test set), the random walk with drift has a non-zero vote probability for all yearly

data. That is, for each yearly series, at least one tree in the random forest voted for random walk with drift.

2. For monthly data, SARIMA, tbats, theta, nn, rwd and stlar have all been assigned a very high vote probability for at least some series.
3. Those monthly series with a high vote probability for SARIMA also have a high vote probability for ETS models with seasonal components. Similarly, those series with a high vote probability for (non-seasonal) ARIMA models also have a high probability for non-seasonal ETS models.
4. Compared to yearly and monthly series, for hourly series the best forecasting model has been selected with very high probability.
5. For hourly series, low vote probabilities have been assigned to random walk, random walk with drift, theta and white noise, while neural networks have been assigned a non-zero vote probability for all hourly series.

We also created an interactive dashboard[1] allowing additional visualizations such as: i) the distribution of true class labels by row clusters of the vote matrix; ii) the feature distribution by clusters; and iii) the distribution of clusters in the feature space (a principal component analysis is used to map each time series as a point in a two-dimensional instance space given by features).

5.2 Which features are the most important and where are they important?

The importance of a feature can be measured using individual conditional expectation (ICE) scores (Friedman, Popescu, et al. 2008; Goldstein et al. 2015) which measure the effect of a change in value of a single feature of a single time series on the output of the algorithm. When averaged across series, these give "partial dependencies". The partial dependencies are often plotted against the feature values to give partial dependency curves (Greenwell, Boehmke & McCarthy 2018).

The partial-dependency score measures the "flatness" of the partial dependence curve for each feature. The flatness of the curve is measured using the standard deviation of the partial dependencies for different values of the feature. A feature with a low partial-dependency score either has little influence on the response variable or the feature interacts with other features.

An ICE score is similar, but is based on measuring the "flatness" of the ICE curves rather than the average ICE curve. The ICE score is the average of the standard deviations of the individual ICE curves. A higher value indicates a higher degree of interactivity with other features.

[1] `https://thiyangt.github.io/fformsviz/fforms.html`

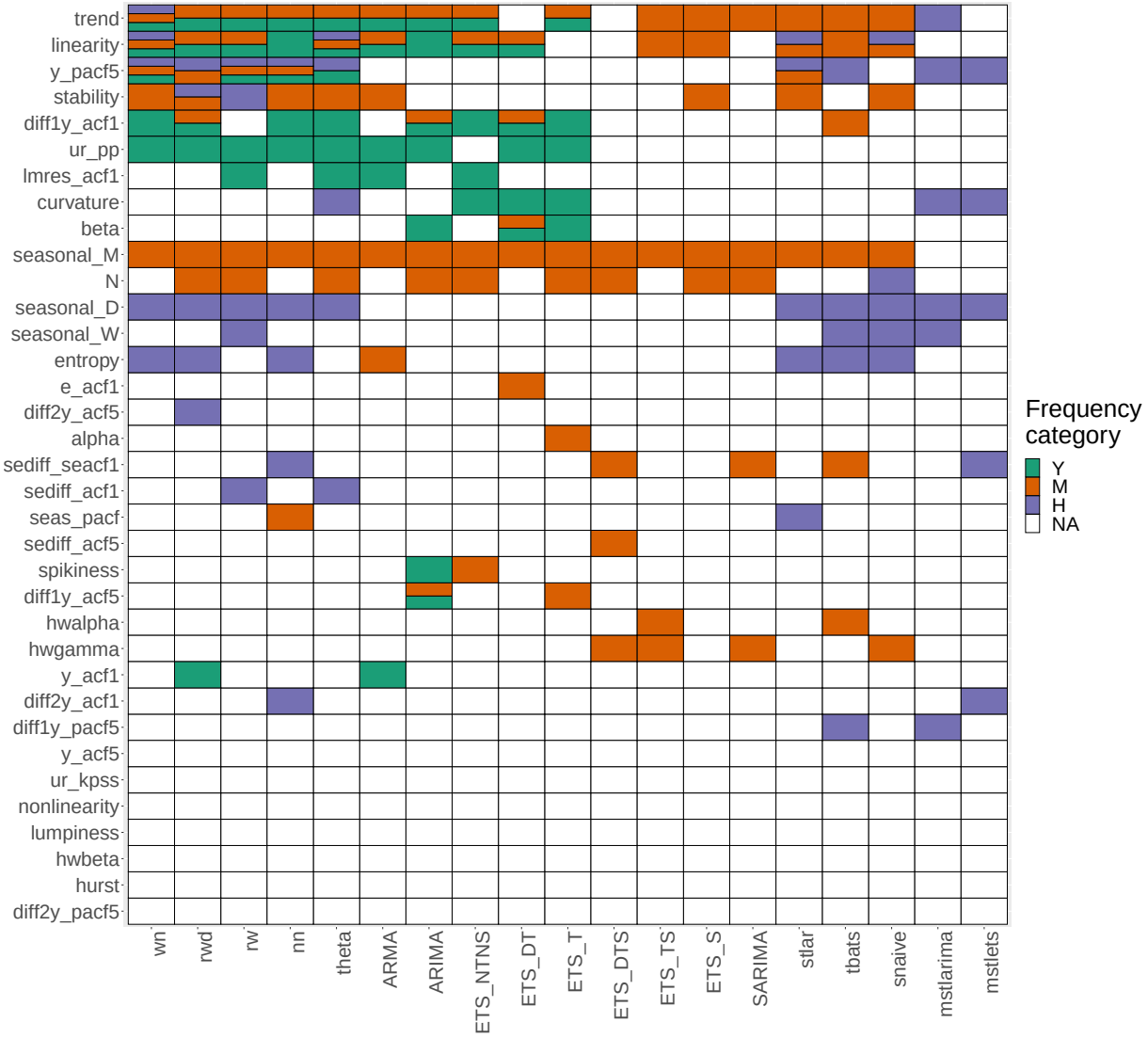

Figure 6: *The top five features for each frequency category and model. The cell colours denote the frequency group (yearly, monthly, hourly), while features are in rows and forecast methods in columns.*

For each feature, an overall rank is calculated based on the mean ranking of three measures of variable importance: i) permutation-based variable importance; ii) the partial-dependency score; and iii) the individual conditional expectation scores.

Figure 6 shows the five most important features (represented by a coloured cell) for each model class across yearly, monthly and hourly series. Note that some columns contain more than five coloured cells for each frequency category as mulitple features were equally ranked. Overall, the most important features are: strength of trend, sum of the first five partial autocorrelation coefficients, stability, the first ACF value of the differenced series and linearity.

For yearly data, the test Phillips-Perron unit root test statistic is also important. Also, features related to different types of trend (beta and curvature) are important in selecting between ETS models.

For monthly data, the length of the time series (T) is assigned a high importance. For all seasonal time series (both monthly and hourly), the strength of seasonality has been selected among the top features across all classes.

For hourly data, the strength of daily seasonality (measured by seasonal_d) appears to be more important than the strength of weekly seasonality (measured by seasonal_w). Entropy and the ACF and PACF features related to seasonal lags and seasonal differencing also rank among the top five for some model classes.

5.3 When and how are features linked to the forecast outcome?

Partial-dependence curves can be used to visualise the relationship between each feature and the choice of forecast model selection (see section Section 5). These show the probability of selection each model as a function of the given feature, while accounting for the average effect of other features in the model. We can compute 95% confidence intervals for these partial dependency estimates, using the ICE curves.

For yearly series, the Phillips-Perron test statistic is the most important feature across many classes. Figure 7 shows the partial dependency curves of the Phillips-Perron test statistic. A high negative value of the Phillips-Perron test statistic indicates a relatively stationary series, while a

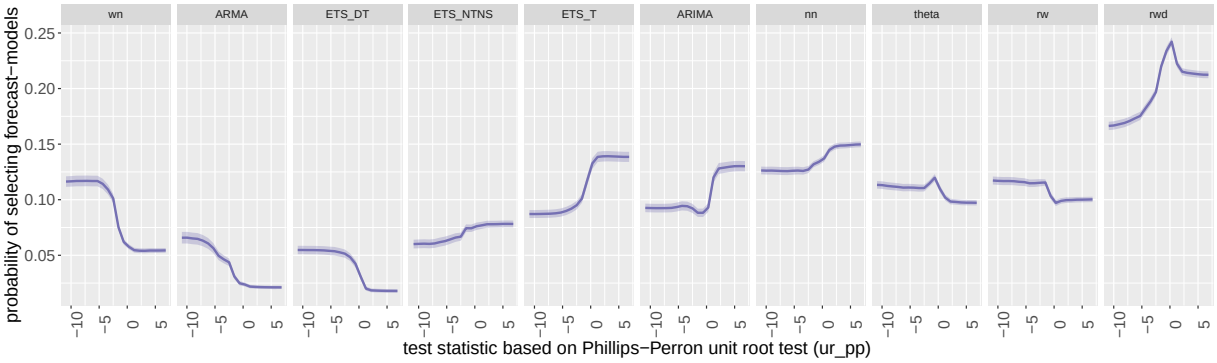

Figure 7: *Partial dependence plots for Phillips-Perron unit root test statisticm, along with 95% confidence intervals. All classes show a turning point in the relationship around zero.*

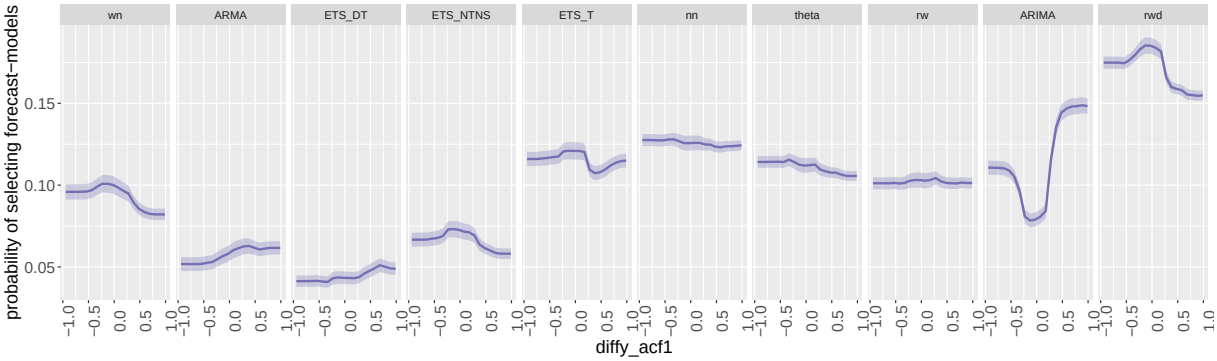

Figure 8: *Partial dependence plots for diff1y_acf1 for yearly series, along with 95% confidence intervals.*

large positive value of the test statistic indicates nonstationarity of the series. As expected, white noise models and stationary ARMA models are chosen for low values of the statistic, while non-stationary ARIMA models and ETS models with trend are selected for high values of the statistic.

The first autocorrelation coefficient of the differenced series (`diff1y_acf1`) is useful in determining the difference stationarity of a time series. This feature has been ranked among the top-five within several classes of yearly series. The associated partial dependency plots are in Figure 8, showing that small absolute values of the feature increases the probability of selection a random walk with drift models, while larger absolute values increase the probability of ARIMA models.

For monthly series, the strength of seasonality is the most important feature across all classes, and Figure 9 shows that the probability of selecting a seasonal forecast model increases as the strength of seasonality increases, while the reverse occurs for non-seasonal models.

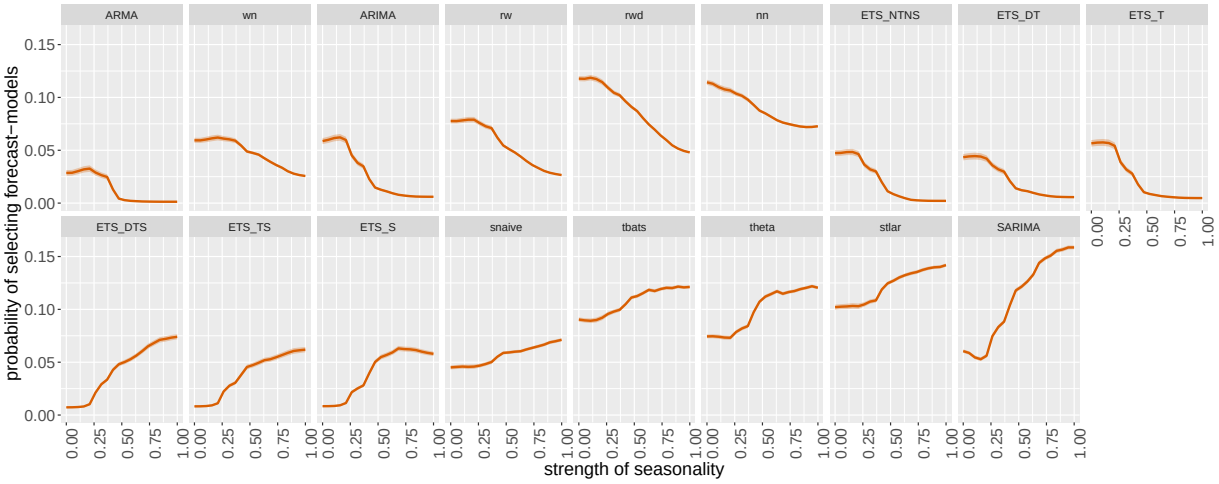

Figure 9: *Partial dependence plots for strength of seasonality for monthly series, along with 95% confidence intervals. The probability of selecting forecast models with seasonal components increases as seasonality increases.*

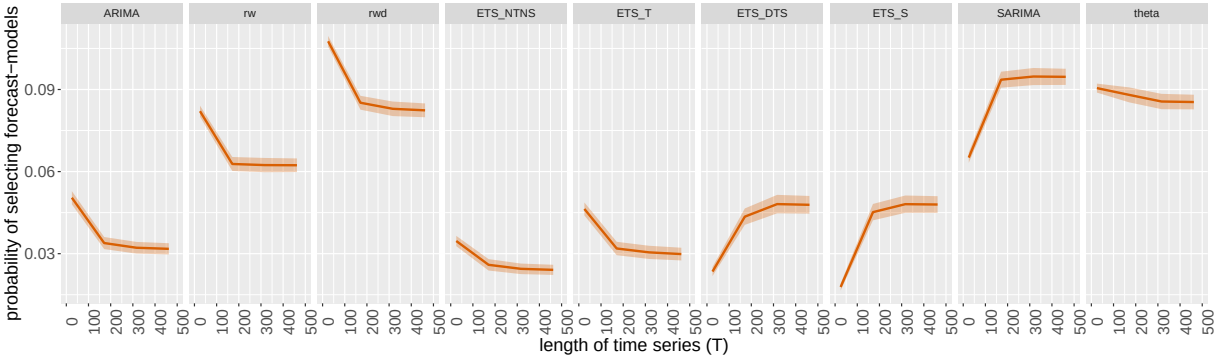

Figure 10: *Partial dependence plots for length of time series (T), along with 95% confidence intervals. The probability of selecting rw and rwd decreases for series with $T > 150$.*

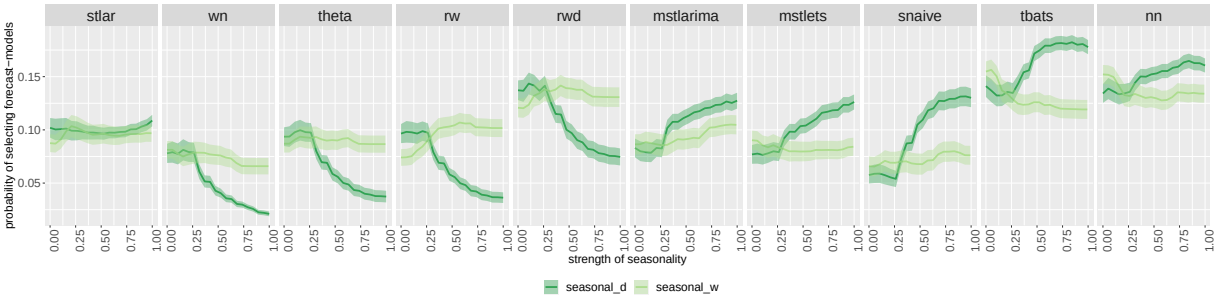

Figure 11: *Partial dependence plots for strength of seasonality for hourly series, along with 95% confidence intervals.*

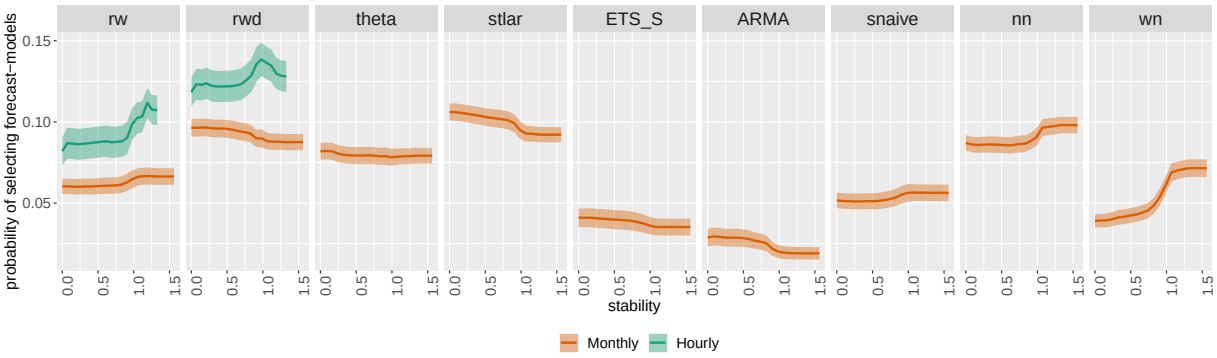

Figure 12: *Partial dependence plots for stability for monthly and hourly series, along with 95% confidence intervals.*

The length of time series is also assigned a high importance within many classes of monthly series. Figure 10 reveals that longer time series lead to the selection of highly parameterized models or models involving seasonal components, while shorter series lead to simple models being selected.

The partial dependency plots of the strength of seasonalities for hourly series, in Figure 11, shows that the probability of selecting simple non-seasonal models such as random walk, random walk with drift, theta and white noise process, decreases as the strength of daily seasonality increases. Weekly seasonality is less important in selecting a model than daily seasonality, probably because daily seasonality tends to be stronger for these series.

For monthly series, stability is ranked among the top five features within many classes, whereas it does not appear in the top five features within any of the classes for yearly series. Further, it is ranked important only within random walk and random walk with drift for hourly series. Figure 12 shows the partial dependency plots for stability, where we can see that the probability of selecting white noise models increases as stability increases for monthly series while for hourly series the probability of selecting random walk with drift and tbats models shows a nonmonotonic structure.

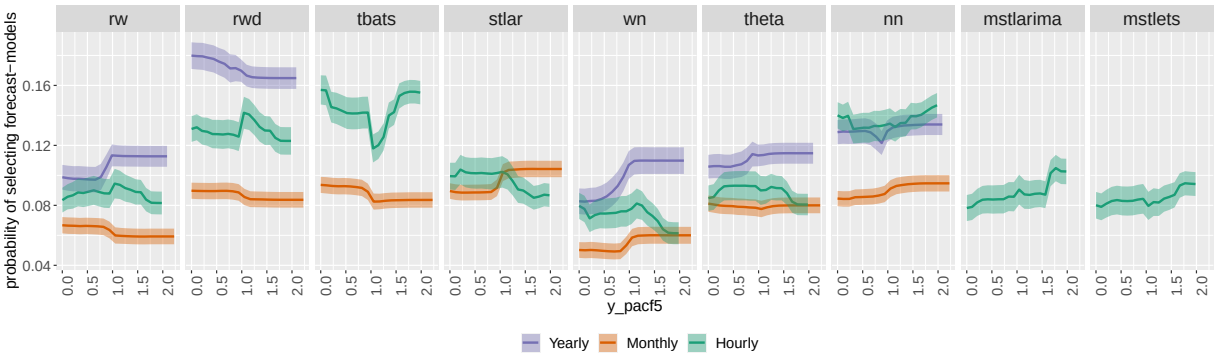

Figure 13: *Partial dependence plots for y_pacf5 for yearly monthly and hourly series, along with 95% confidence intervals.*

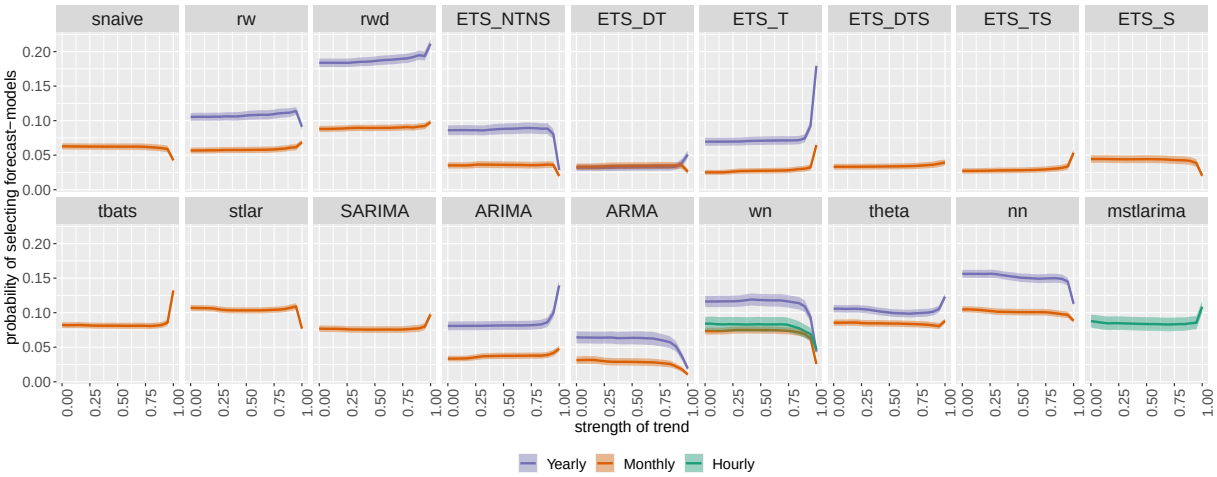

Figure 14: *Partial dependence plots for trend for yearly and monthly series, along with 95% confidence intervals. The probability of selecting ETS models without a trend component and stationary models (WN and ARMA) decreases for an extremely high value of trend.*

The sum of the first five partial autocorrelation coefficients, strength of trend and linearity are the most commonly selected features within many classes across all three frequency categories. The partial dependency plots for the sum of the first five partial autocorrelation coefficients are shown in Figure 13, showing that the relationship varies across classes as well as frequency categories.

Figure 14 show the partial dependency curves for strength of trend. The probability of selecting a white noise, stationary ARMA or ETS model without trend components, decreases as the strength of trend increases, while the opposite relationship can be observed for other classes.

The partial dependency plot for linearity in Figure 15 shows that for yearly series, the random walk with drift and ARMA classes are highly sensitive to the value of linearity around 0. However, the partial dependency curves of linearity for other classes are relatively flat throughout the range. This could be due to the interaction of linearity with other features.

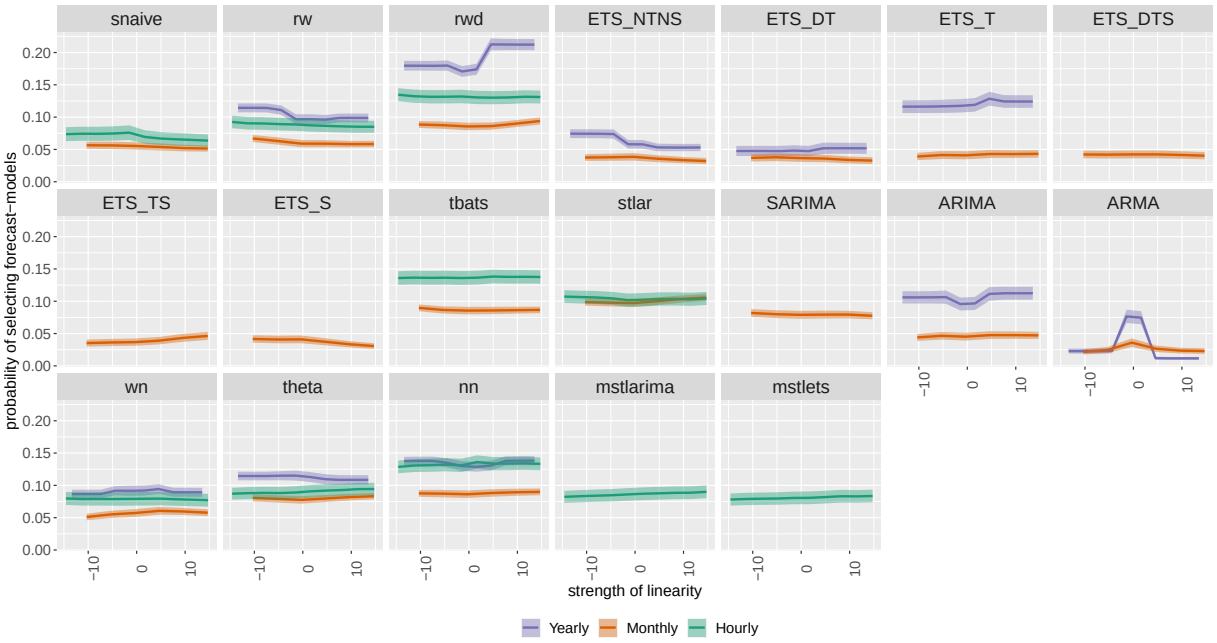

Figure 15: *Partial dependence plots for linearity for yearly monthly and hourly series, along with 95% confidence intervals.*

The associated two-way partial dependency plots for linearity are shown in Figures 16–18 for yearly, monthly and hourly series respectively. These also show the probability of selecting each model, but over two variables (holding the others at average values), allowing us to see interactions between features.

Figure 16 shows a pattern of interactivity between linearity and both lmres_acf1 and diff1y_acf1, within the wn, rw, rwd, theta and nn model classes. For monthly series (Figure 17), linearity shows interaction with the ACF value at the first seasonal lag of seasonally-differenced series, stability, the first ACF value at the remainder series and a parameter estimate of the ETS(A,A,A) model. Figure 18 shows the two-way partial dependency plot for hourly series between linearity

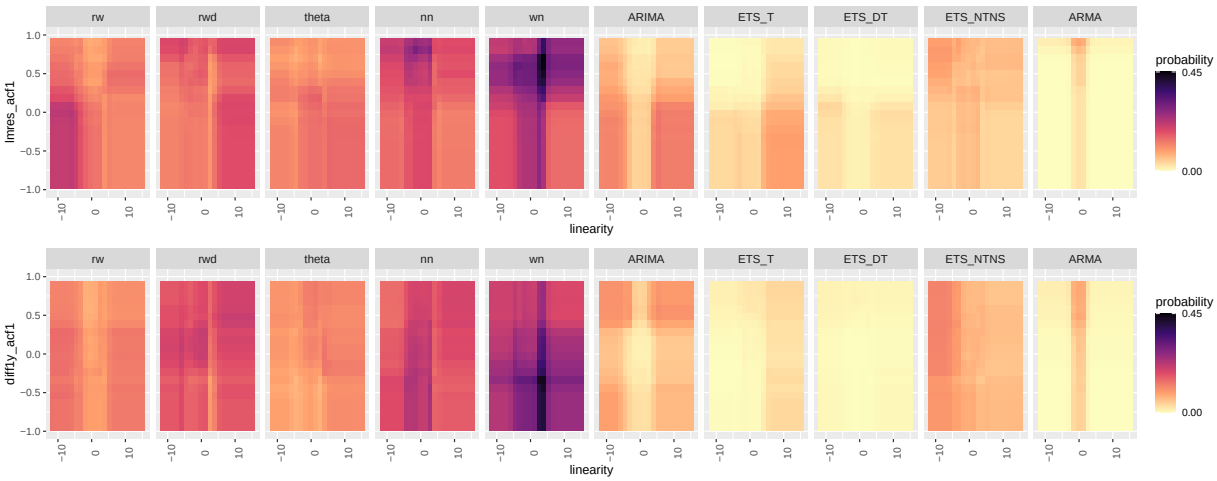

Figure 16: *Two-way partial dependence plots for linearity for yearly series.*

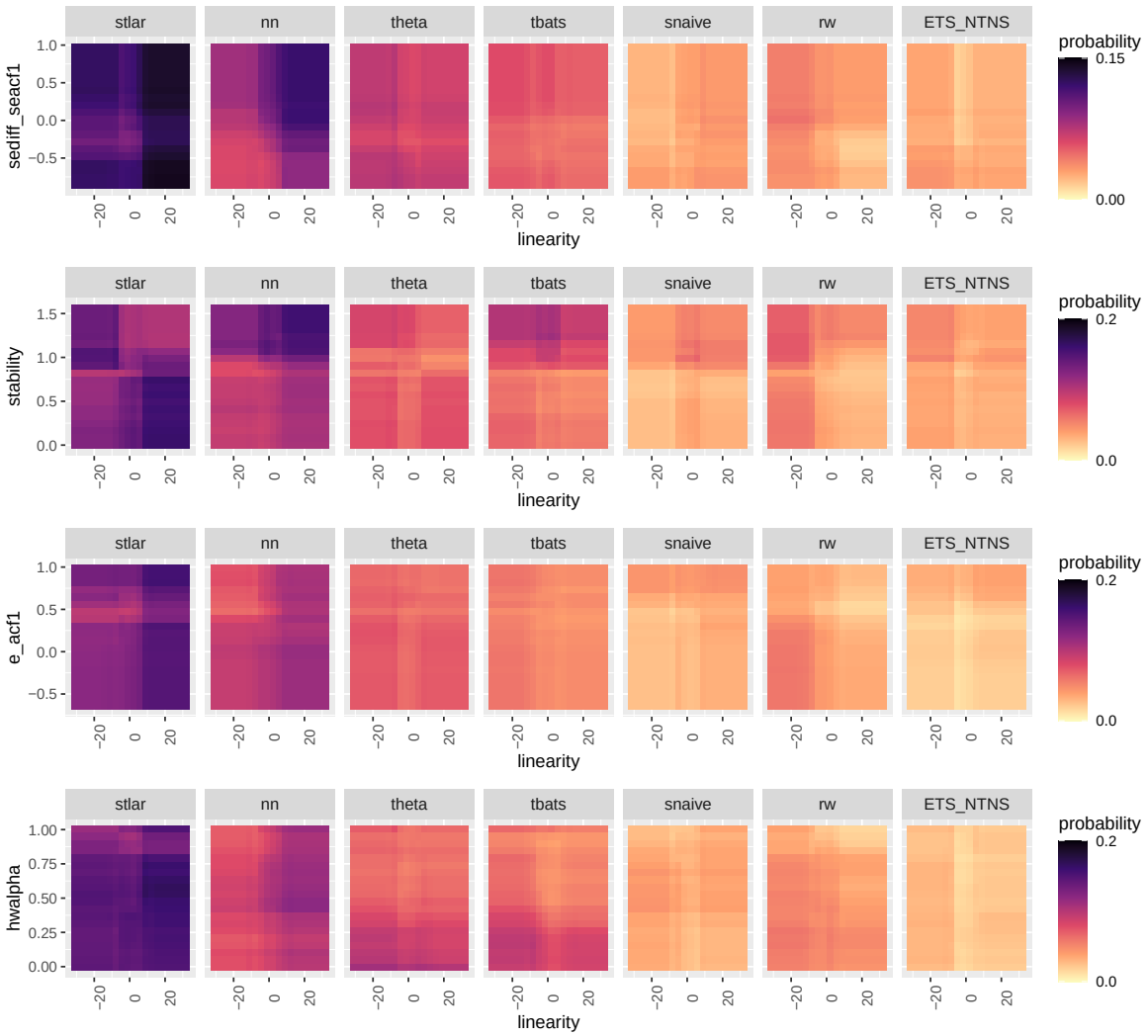

Figure 17: *Two-way partial dependence plots for linearity for monthly series.*

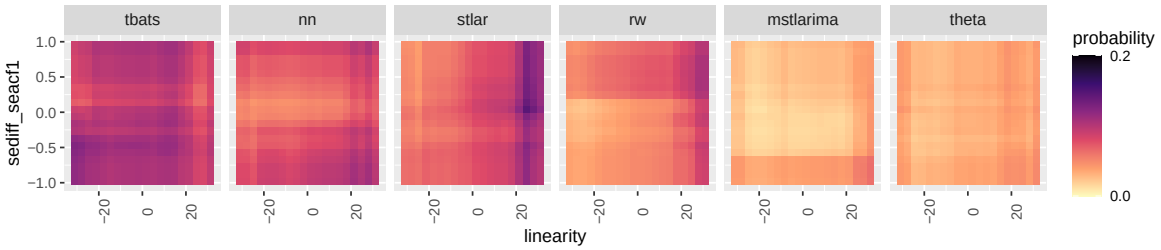

Figure 18: *Two-way partial dependence plots for linearity for hourly series.*

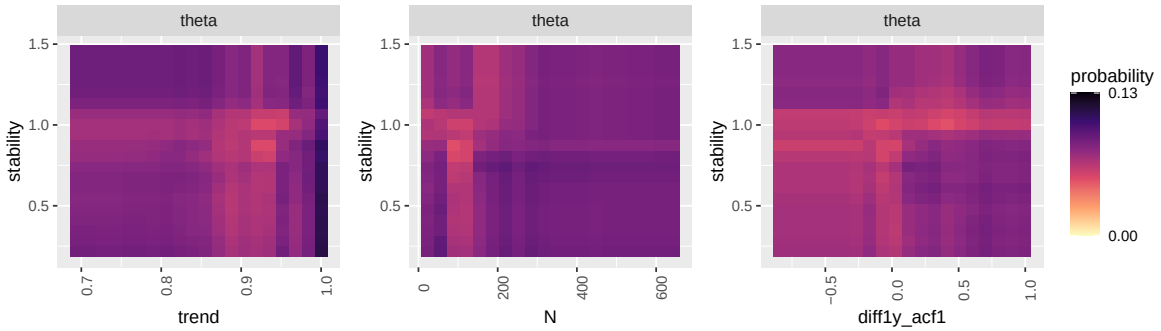

Figure 19: *Two-way partial dependence plots for linearity for monthly series within the theta class.*

and the ACF value at the first seasonal lag of seasonally-differenced series. Within rw and mstlarima we can see a separation between the lower half and the upper half due to the main effect of sediff_seacf1.

Figure 19 shows how stability interacts with trend, length, and first autocorrelation coefficient of the differenced series. Although the partial dependency curve of stability for the class theta is relatively flat (Figure 12), Figure 19 shows patterns of interaction between stability and other features.

6 Discussion and conclusions

We have proposed a novel framework for forecast-model selection using meta-learning based on time series features. Our proposed FFORMS algorithm uses the knowledge of the past performance of candidate forecast models on a collection of time series in order to identify suitable forecast models for new series. A key advantage of FFORMS is the idea of outsourcing most of the heavy computational work to the offline phase. Therefore, unlike traditional model selection strategies or cross-validation processes, our proposed framework can be used to produce relatively accurate and fast forecasts for very large collections of time series. For real-time forecasting, our framework involves only the calculation of features, the selection of a forecast method based on FFORMS random forest classifier, and the calculation of the forecasts from the chosen model. None of these steps involve substantial computation, and they can be easily parallelized when forecasting for a large number of new time series. In doing so, the FFORMS framework fills an important gap in contemporary forecasting practice, with many available models to choose from and with forecasts being required in a short time. We are well aware that the FFORMS forecasts will not usually be the best possible forecasts given unlimited computation time; rather, they are almost as good but available in a much shorter time period. Trading off computational time for accuracy, Montero-Manso et al. (2020) and Talagala, Li & Kang (2022) build on FFORMS to introduce FFORMA and FFORMPP.

This paper has made a first step towards providing a comprehensive analysis of the relationship between time series features and forecast model selection using machine learning interpretability techniques. We have explored the role of features in forecast model selection in the FFORMS framework. First we explored the similarities between series learned by the random forest. Next, we explored **which** features are the most important for model selection in the FFORMS framework, and where they are most important. Features such as strength of trend, strength of seasonality, linearity, information related to partial-autocorrelation structure, features related

to difference stationary, length and entropy are the most important features. Finally, partial dependency plots were used to visualise **when** and **how** these features link with the prediction outcome of the FFORMS framework. This improves our understanding of the data and the relationship learned by the random forest algorithm, and increases trust in the results obtained from the proposed framework. Our method is sensitive to the lengths of the time series because we employed length as an input feature. We found that whereas longer time series prefer to choose more parameterized models, shorter time series are more likely to choose simple models like random walk, white noise, etc. However, PDP curves of length revealed that there is a threshold beyond which the probability of choosing various models in response to training set length starts to plateau.

Hyperparameter tuning, feature selection, and feature engineering are essential tasks in machine learning that can have a significant effect on the performance of the machine learning algorithm. However, in order to keep the algorithm simple and lower the cost of the tuning process for the offline phase of the algorithm, hyperparameter tuning, feature selection, and engineering were not carried out in this study. Due to the large and complex search space, feature selection and engineering are not conducted. The stochastic nature of the meta-learning algorithm makes parameter tuning a challenging issue because each parameter vector must be evaluated numerous times. However, we believe both hyperparameter tuning and feature selection and engineering are intriguing future topics to study.

One could argue machine learning-based forecasting models such as NNs, have a stochastic nature, which translates into different results for the same input data in different training sessions. This may have a negative impact on the choice of the best model. While this problem is difficult to overcome, one could fit such models multiple times with different random seeds and use the average of the forecast error measure to select best forecast model when labelling time series.

Future research in this area could consider how local interpretable model-agnostic explanations can be used to identify regions of data where features contribute most to the classification of a specific instance.

The features and models we have used in our analyses may be varied for other applications. Features can be added or removed as required, and new models can be introduced that are appropriate to the application and types of time series being forecast. While some features will probably be useful across a broad range of problems, it is likely that some features will only be useful for some specific types of data or applications.

Supplementary Materials

Reproducibility: All the code and data used in this paper is available at `https://github.com/thiyangt/fforms`. The R packages `ggplot2` (Hadley 2009) and `iheatmapr` (Schep & Kummerfeld 2017) provides the basis for graphics. The document is written in `knitr` (Xie 2017).

Software: The R package `seer` implements the FFORMS algorithm and is available on CRAN (`https://cran.r-project.org/web/packages/seer/index.html`).

Interactive dashboard visualisation: A shiny app demonstrating the vote matrices is available online at `https://thiyangt.github.io/fformsviz/fforms.html`

Appendix: Definition of features

Length of time series

The appropriate forecast method for a time series depends on how many observations are available. For example, shorter series tend to need simple models such as a random walk. On the other hand, for longer time series, we have enough information to be able to estimate a number of parameters. For even longer series (over 200 observations), models with time-varying parameters give good forecasts as they help to capture the changes of the model over time.

Features based on an STL-decomposition

The strength of trend, strength of seasonality, linearity, curvature, spikiness and first autocorrelation coefficient of the remainder series, are calculated based on a decomposition of the time series. Suppose we denote our time series as $y_1, y_2, \dots, y_T$. First, an automated Box-Cox transformation (Guerrero 1993) is applied to the time series in order to stabilize the variance and to make the seasonal effect additive. The transformed series is denoted by y_t^*. For quarterly and monthly data, this is decomposed using MSTL (Bandara, Hyndman & Bergmeir 2021) to give $y_t^* = T_t + S_t + R_t$, where T_t denotes the trend, S_t denotes one or more seasonal components, and R_t is the remainder component. For non-seasonal data, Friedman's super smoother (Friedman 1984) is used to decompose $y_t^* = T_t + R_t$, and $S_t = 0$ for all t. The de-trended series is $y_t^* - T_t = S_t + R_t$, the deseasonalized series is $y_t^* - S_t = T_t + R_t$.

The strength of trend is measured by comparing the variance of the deseasonalized series and the remainder series (Wang, Smith-Miles & Hyndman 2009):

$$\text{Trend} = \max\left[0, 1 - \text{Var}(R_t)/\text{Var}(T_t + R_t)\right].$$

Similarly, the strength of seasonality is computed as

$$\text{Seasonality} = \max\left[0, 1 - \text{Var}(R_t)/\text{Var}(S_t + R_t)\right].$$

The linearity and curvature features are based on the coefficients of an orthogonal quadratic regression

$$T_t + R_t = \beta_0 + \beta_1\phi_1(t) + \beta_2\phi_2(t) + \varepsilon_t,$$

where $t = 1, 2, \dots, T$, and ϕ_1 and ϕ_2 are orthogonal polynomials of orders 1 and 2. The estimated value of β_1 is used as a measure of linearity while the estimated value of β_2 is considered as a measure of curvature. These features were used by Hyndman, Wang & Laptev (2015). The

linearity and curvature depend on the the scale of the time series. Therefore, the time series are scaled to mean zero and variance one before these two features are computed.

The spikiness feature is useful when a time series is affected by occasional outliers. Hyndman, Wang & Laptev (2015) introduced an index to measure spikiness, computed as the variance of the leave-one-out variances of R_t. We compute the first autocorrelation coefficient of the remainder series, R_t.

Stability and lumpiness

The features "stability" and "lumpiness" are calculated based on tiled windows (i.e., they do not overlap). For each window, the sample mean and variance are calculated. The stability feature is calculated as the variance of the means, while lumpiness is the variance of the variances. These were first used by Hyndman, Wang & Laptev (2015).

Spectral entropy of a time series

Spectral entropy is based on information theory, and can be used as a measure of the forecastability of a time series. Series that are easy to forecast should have a small spectral entropy value, while very noisy series will have a large spectral entropy. We use the measure introduced by Goerg (2013) to estimate the spectral entropy. It estimates the Shannon entropy of the spectral density of the normalized spectral density, given by

$$H_s(y_t) = -\int_{-\pi}^{\pi} \hat{f}_y(\lambda) \log \hat{f}_y(\lambda) d\lambda,$$

where $\hat{f}_y$ denotes the estimate of the spectral density introduced by Nuttall & Carter (1982). The R package ForeCA (Goerg 2020) was used to compute this measure.

Hurst exponent

The Hurst exponent measures the long-term memory of a time series. The Hurst exponent is given by $H = d + 0.5$, where d is the fractal dimension obtained by estimating an ARFIMA$(0, d, 0)$ model. We compute this using the maximum likelihood method (Haslett & Raftery 1989) as implemented in the fracdiff package (Fraley 2020). This measure was also used in Wang, Smith-Miles & Hyndman (2009).

Nonlinearity

To measure the degree of nonlinearity of the time series, we use the statistic computed in Terasvirta's neural network test for nonlinearity (Teräsvirta, Lin & Granger 1993), also used in

Wang, Smith-Miles & Hyndman (2009). This takes large values when the series is nonlinear, and values around 0 when the series is linear.

Parameter estimates of an ETS model

The ETS (A,A,N) model (Hyndman et al. 2008) produces equivalent forecasts to Holt's linear trend method, and can be expressed as follows:

$$\begin{aligned} y_t &= \ell_{t-1} + b_{t-1} + \varepsilon_t \\ \ell_t &= \ell_{t-1} + b_{t-1} + \alpha\varepsilon_t \\ b_t &= b_{t-1} + \beta\varepsilon_t, \end{aligned}$$

where α is the smoothing parameter for the level, and β is the smoothing parameter for the trend. We include the parameter estimates of both α and β in our feature set for yearly time series. These indicate the variability in the level and slope of the time series.

The ETS (A,A,A) model (Hyndman et al. 2008) produces equivalent forecasts to Holt-Winters' additive method, and can be expressed as follows:

$$\begin{aligned} y_t &= \ell_{t-1} + b_{t-1} + s_{t-m} + \varepsilon_t \\ \ell_t &= \ell_{t-1} + b_{t-1} + s_{t-m} + \alpha\varepsilon_t \\ b_t &= b_{t-1} + \beta\varepsilon_t, \\ s_t &= s_{t-m} + \gamma\varepsilon_t, \end{aligned}$$

where γ is the smoothing parameter for the seasonal component, and the other parameters are as above. We include the parameter estimates of α, β and γ in our feature set for monthly and quarterly time series. The value of γ provides a measure for the variability of the seasonality of a time series.

Unit root test statistics

The Phillips-Perron test is based on the regression $y_t = c + \alpha y_{t-1} + \varepsilon_t$. The feature is the usual "Z-alpha" statistic with the Bartlett window parameter set to the integer value of $4(T/100)^{0.25}$ (Pfaff 2008). This is the default value returned from the `ur.pp()` function in the `urca` package (Pfaff, Zivot & Stigler 2016).

The KPSS test is based on the regression $y_t = c + \delta t + \alpha y_{t-1} + \varepsilon_t$. The feature is the usual KPSS statistic with the Bartlett window parameter set to the integer value of $4(T/100)^{0.25}$ (Pfaff 2008).

This is the default value returned from the `ur.kpss()` function in the `urca` package (Pfaff, Zivot & Stigler 2016).

Autocorrelation coefficient based features

We calculate the first-order autocorrelation coefficient and the sum of squares of the first five autocorrelation coefficients of the original series, the first-differenced series, the second-differenced series, and the seasonally differenced series (for seasonal data). A linear trend model is fitted to the time series, and the first-order autocorrelation coefficient of the residual series is calculated. We calculate the sum of squares of the first five partial autocorrelation coefficients of the original series, the first-differenced series and the second-differenced series.

References

Armstrong, JS (2001). Should we redesign forecasting competitions? *International Journal of Forecasting* **17**(1), 542–543.

Bandara, K, RJ Hyndman & C Bergmeir (2021). MSTL: A Seasonal-Trend Decomposition Algorithm for Time Series with Multiple Seasonal Patterns. *arXiv preprint 2107.13462*. https://arxiv.org/abs/2107.13462.

Breiman, L (2001). Random forests. *Machine Learning* **45**(1), 5–32.

Breiman, L & A Cutler (2004). *Random Forests*. Version 5.1. https://www.stat.berkeley.edu/~breiman/RandomForests/.

Breiman, L, A Cutler, A Liaw & M Wiener (2018). *randomForest: Breiman and Cutler's Random Forests for Classification and Regression*. R package version 4.6-14. https://cran.r-project.org/web/packages/randomForest/.

Chen, C, A Liaw & L Breiman (2004). *Using random forest to learn imbalanced data*. Tech. rep. University of California, Berkeley. http://statistics.berkeley.edu/sites/default/files/tech-reports/666.pdf.

Collopy, F & JS Armstrong (1992). Rule-based forecasting: development and validation of an expert systems approach to combining time series extrapolations. *Management Science* **38**(10), 1394–1414.

Fraley, C (2020). *fracdiff: Fractionally differenced ARIMA aka ARFIMA(p,d,q) models*. R package version 1.5-1. https://CRAN.R-project.org/package=fracdiff.

Friedman, JH (1984). *A variable span scatterplot smoother*. Technical Report 5. Laboratory for Computational Statistics, Stanford University.

Friedman, JH, BE Popescu, et al. (2008). Predictive learning via rule ensembles. *The Annals of Applied Statistics* **2**(3), 916–954.

Fulcher, BD & NS Jones (2014). Highly comparative feature-based time-series classification. *IEEE Transactions on Knowledge and Data Engineering* **26**(12), 3026–3037.

Goerg, GM (2020). *ForeCA: An R package for forecastable component analysis*. R package version 0.2.7. https://CRAN.R-project.org/package=ForeCA.

Goerg, G (2013). Forecastable component analysis. In: *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pp.64–72.

Goldstein, A, A Kapelner, J Bleich & E Pitkin (2015). Peeking inside the black box: Visualizing statistical learning with plots of individual conditional expectation. *Journal of Computational and Graphical Statistics* **24**(1), 44–65.

Greenwell, BM, BC Boehmke & AJ McCarthy (2018). A Simple and effective model-based variable importance measure. *arXiv preprint arXiv: 1805.04755*.

Guerrero, VM (1993). Time-series analysis supported by power transformations. *Journal of Forecasting* **12**, 37–48.

Hadley, W (2009). ggplot2: Elegant graphics for data analysis. *New York: Springer*.

Haslett, J & AE Raftery (1989). Space-time modelling with long-memory dependence: Assessing Ireland's wind power resource. *Applied Statistics* **38**(1), 1–50.

Hyndman, R, G Athanasopoulos, C Bergmeir, G Caceres, L Chhay, M O'Hara-Wild, F Petropoulos, S Razbash, E Wang & F Yasmeen (2021). *forecast: Forecasting functions for time series and linear models*. R package version 8.15. http://pkg.robjhyndman.com/forecast.

Hyndman, RJ (2001). It's time to move from what to why. *International Journal of Forecasting* **17**(1), 567–570.

Hyndman, RJ & G Athanasopoulos (2018). *Forecasting: principles and practice*. 2nd ed. Melbourne, Australia: OTexts. https://OTexts.org/fpp2/.

Hyndman, RJ & Y Khandakar (2008). Automatic time series forecasting: the forecast package for R. *Journal of Statistical Software* **26**(3), 1–22. http://www.jstatsoft.org/article/view/v027i03.

Hyndman, RJ, AB Koehler, JK Ord & RD Snyder (2008). *Forecasting with exponential smoothing: the state space approach*. Berlin: Springer.

Hyndman, RJ, AB Koehler, RD Snyder & S Grose (2002). A state space framework for automatic forecasting using exponential smoothing methods. *International Journal of Forecasting* **18**(3), 439–454.

Hyndman, RJ, E Wang & N Laptev (2015). Large-scale unusual time series detection. In: *Data Mining Workshop (ICDMW), 2015 IEEE International Conference on*. IEEE, pp.1616–1619.

Kalousis, A & T Theoharis (1999). NOEMON: Design, implementation and performance results of an intelligent assistant for classifier selection. *Intelligent Data Analysis* **3**(5), 319–337.

Kang, Y, RJ Hyndman & F Li (2020). GRATIS: GeneRAting TIme Series with diverse and controllable characteristics. *Statistical Analysis and Data Mining: The ASA Data Science Journal* **13**(4), 354–376.

Kang, Y, F Li, RJ Hyndman, M O'Hara-Wild & B Zhao (2020). *gratis: GeneRAting TIme Series with diverse and controllable characteristics*. R package version 0.2.1. https://CRAN.R-project.org/package=gratis.

Kück, M, SF Crone & M Freitag (2016). Meta-learning with neural networks and landmarking for forecasting model selection an empirical evaluation of different feature sets applied

to industry data. In: *Neural Networks (IJCNN), 2016 International Joint Conference on*. IEEE, pp.1499–1506.

Lawrence, M (2001). Why another study? *International Journal of Forecasting* **17**(1), 574–575.

Lemke, C & B Gabrys (2010). Meta-learning for time series forecasting and forecast combination. *Neurocomputing* **73**(10), 2006–2016.

Liaw, A & M Wiener (2002). Classification and regression by randomForest. *R News* **2**(3), 18–22. http://CRAN.R-project.org/doc/Rnews/.

Makridakis, S, A Andersen, R Carbone, R Fildes, M Hibon, R Lewandowski, J Newton, E Parzen & R Winkler (1982). The accuracy of extrapolation (time series) methods: results of a forecasting competition. *Journal of forecasting* **1**(2), 111–153.

Makridakis, S & M Hibon (2000). The M3-Competition: results, conclusions and implications. *International Journal of Forecasting* **16**(4), 451–476.

Makridakis, S, E Spiliotis & V Assimakopoulos (2019). The M4 Competition: 100,000 time series and 61 forecasting methods. *International Journal of Forecasting*.

Mersmann, O (2021). *microbenchmark: Accurate Timing Functions*. R package version 1.4.9. https://CRAN.R-project.org/package=microbenchmark.

Montero-Manso, P, G Athanasopoulos, RJ Hyndman & TS Talagala (2020). FFORMA: Feature-based forecast model averaging. *International Journal of Forecasting* **36**(1), 86–92. https://doi.org/10.1016/j.ijforecast.2019.02.011.

Nuttall, AH & GC Carter (1982). Spectral estimation using combined time and lag weighting. *Proceedings of the IEEE* **70**(9), 1115–1125.

Pfaff, B (2008). *Analysis of integrated and cointegrated time series with R*. Springer.

Pfaff, B, E Zivot & M Stigler (2016). *urca: Unit root and cointegration tests for time series data*. R package version 1.3-0. https://CRAN.R-project.org/package=urca.

Prudêncio, RB & TB Ludermir (2004). Meta-learning approaches to selecting time series models. *Neurocomputing* **61**, 121–137.

R Core Team (2018). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing. Vienna, Austria. https://www.R-project.org/.

Racine, J (2000). Consistent cross-validatory model-selection for dependent data: hv-block cross-validation. *Journal of Econometrics* **99**(1), 39–61.

Reid, DJ (1972). "A comparison of forecasting techniques on economic time series". In: *Forecasting in Action*. Birmingham, UK: Operational Research Society.

Rice, JR (1976). The algorithm selection problem. *Advances in Computers* **15**, 65–118.

Schep, AN & SK Kummerfeld (2017). iheatmapr: Interactive complex heatmaps in R. *Journal of Open Source Software*. http://dx.doi.org/10.21105/joss.00359.

Shah, C (1997). Model selection in univariate time series forecasting using discriminant analysis. *International Journal of Forecasting* **13**(4), 489–500.

Smith-Miles, K (2009). Cross-disciplinary perspectives on meta-learning for algorithm selection. *ACM Computing Surveys (CSUR)* **41**(1), 6.

Talagala, TS, F Li & Y Kang (2022). FFORMPP: Feature-based forecast model performance prediction. *International Journal of Forecasting* (forthcoming). https://doi.org/10.1016/j.ijforecast.2021.07.002.

Teräsvirta, T, CF Lin & CWJ Granger (1993). Power of the neural network linearity test. *Journal of Time Series Analysis* **14** (2), 209–220.

Wang, X, K Smith-Miles & RJ Hyndman (2009). Rule induction for forecasting method selection: meta-learning the characteristics of univariate time series. *Neurocomputing* **72**(10), 2581–2594.

Widodo, A & I Budi (2013). "Model selection using dimensionality reduction of time series characteristics". Paper presented at the International Symposium on Forecasting, Seoul, South Korea. June 2013. https://goo.gl/ig2J57.

Xie, Y (2017). *Dynamic Documents with R and knitr*. Chapman and Hall/CRC.